AD 658472

SP-2944/000/00

THE MADAM SYSTEM:

DATA MANAGEMENT WITH A SMALL COMPUTER

E. W. Franks

8 September 1967

19

SP-2944/000/00

SP *a professional paper*

THE MADAM SYSTEM:

DATA MANAGEMENT WITH A SMALL COMPUTER

by

E. W. Franks

8 September 1967

SYSTEM

DEVELOPMENT

CORPORATION

2500 COLORADO AVE.

SANTA MONICA

CALIFORNIA

90406

A-1162 REV. 4 65

ABSTRACT

This paper discusses the uses of the MADAM system for data management tasks on the small, free-standing computer such as the IBM 1401 or IBM 360/30. It is largely nontechnical in approach and is addressed to managers rather than to computer personnel. An example of a problem and its solution is given to demonstrate the speed and simplicity of MADAM usage.

How Big is Small?

Depending upon their backgrounds, people are likely to have different definitions of what constitutes a small computer. I have heard one expert describe an IBM 360 Model 50 with 128K core and several disc packs as a small computer. Others feel that anything beyond a desk-side engineering computer cannot be classified as small. With this wide range of views in mind, I will define the small computer, for purposes of this paper, as the smallest on which it is practicable to operate a data management system. It must have enough high speed memory to house a data management program together with the unit records for input and output. It must have sufficient input/output capability--at least four magnetic storage devices--to perform operations of sorting and merging which are essential to data management. The instruction set must provide arithmetic and logical capability. These criteria eliminate many small computers which are not practical for data management system use. Nevertheless, they leave within the defined class a large number of machines representing almost every significant manufacturer. A good example of the class is the medium-to-small business computer. The particular data management system I am about to describe, MADAM (for Moderately Advanced Data Management), has been prepared for two such machines.

It might be instructive to consider the reasoning behind the selection of the original computer for which the MADAM system was first implemented. This machine, the IBM 1401, has the following minimum features:

Feature	Comment
8K of memory	Less than 8K does not provide adequate storage for management programs and significant data.
Advanced Programming Features	Indexing and high-low-equal compare make possible the coding of generalized management programs compactly and efficiently.
4 Tape Units	This is the minimum for accommodating input and output files and a system master tape and for efficient file sorting.

What is "Data Management"?

In the B.C. (Before Computers) era the functions we now call data management consisted of keeping books and maintaining files by hand. Books had to be accurate and up to date. Files had to be organized so that information stored in them could be retrieved as needed. New information had to be added and obsolete information purged. These requirements held true whether the environment was business, government, academic research or a library. Today nothing has changed except the means of accomplishment. Thus a

computerized data management system is one which can keep accurate and timely records and can build, maintain and retrieve information from files.

Why Generalized Systems?

Every organization has its own methods of record keeping, its own set of periodic reports and its own requirements for file organization. Most organizations that have computerized these methods have written or procured computer programs tailored to their needs. Nobody else can use these programs. Moreover, the particular organization cannot change or expand its methods without writing new programs or modifying old ones. The writing of programs and the updating of existing programs is extremely expensive and tends to inhibit full use of a computer's potential by an organization. Fortunately, however, despite the individual characteristics of each organization's method of operation, all such operations, considered broadly, closely resemble one another. The functions which are performed are few in number and standard in application. The functions themselves are independent of data content and of data format. It is possible, therefore, to produce a set of programs that contain capabilities for the standard functions of data management. All that is required is to specify at execution time the data formats involved and the rules for decisions which must be made on the basis of data content. Such specifications do not have to be made in the language of computers; they can be made in the language of the bookkeeper, a file clerk, librarian, or data analyst, a language with which the data handler is thoroughly familiar and in the use of which he is less likely to make expensive errors than is a programmer using computer code. Such program systems are possible and economic pressure has resulted in many efforts to construct them. Most such efforts, however, are designed for use on large computers. The user who has most economic need for an off-the-shelf system, the small computer user, is largely neglected. MADAM is an attempt to remedy that neglect.

Characteristics of the MADAM System

The MADAM system is currently available on the IBM 1400 series machines and the IBM 360/30. No attempt is made in this paper to differentiate between the two versions in the discussion of characteristics and capabilities. Detailed information about each is found in some of the documents listed in the bibliography.

In view of some of the publicity currently surrounding the glamorous data management systems being built for large computers, it would be useful at the start to make some disclaimers for MADAM, to state what it is not. In the first place it is not an on-line system. Inputs go in and outputs are returned via the computer operating console area. In the second place it is not a symbolic data manipulator. Data elements are not, as in most systems, first defined and named and then referred to by name. As far as the first restriction is concerned, the designers of MADAM feel that a vast

amount of data management work does not require nor justify the expense of an on-line system, especially in a small installation. In the second matter, data symbology has been sacrificed for efficiency. It is the capability for defining symbols that makes so many systems too large for the small machine and which makes small-machine assemblies take seven or more passes to convert symbolic program code to machine format. MADAM is a translate-and-go system in which the interval between translation and execution is scarcely perceptible to the observing layman.

The MADAM system consists of a control program and a series of translator-interpreter pairs of programs which are called in response to the various commands in the MADAM language. The MADAM system resides on a magnetic tape, and the specifications which control its operation are entered through the card reader or from a MADAM library tape. MADAM continues to operate, performing one set of specifications at a time, until no more specifications are left to execute. It is usual to enter a number of such sets whose operation constitute a complete data management job.

User-Oriented Language

The term user-oriented is very much in vogue, and is claimed to apply to the control language of most systems produced today. Not all users are alike, however, and a language oriented to a so-to-speak, least common denominator user would scarcely be satisfactory to the majority of all possible users. User-oriented must be understood as oriented to the user for which the system is intended. Earlier in this document I used the phrase data-handler, and I believe this term comes as close as possible to describing the user for whom the MADAM language is user-oriented. This user is not someone picked casually off the street. On the other hand, he is not the trained professional, the scientist, the engineer or the senior business executive. Such people may, and indeed do, make use of MADAM from time to time. The intended user, however, is the one who has daily contact with the data managed by the system, who is responsible for entering data into the system and for producing the reports generated by the system. Such a user is familiar with the details of data format and data content, and for him the MADAM language gives an easy-to-use, rather natural method of controlling the computerized data management application. This person is seldom the ultimate user of the system, but he is the necessary interface between those who need the output or supply the input, and the computerized facility.

Procedural and Non-Procedural Specifications

MADAM handles two kinds of specifications presented to it in the MADAM language. Procedural specifications are those in which the system is instructed on a step-by-step basis how to process the data: that is, when to read, what to transfer, what to compute, what to compare, what to print and so forth. With this type of specification the user has great flexibility in producing formatted reports, in selecting or merging data files or updating a data base.

The second kind of specification is called non-procedural. Here the operation is standard and highly automatic, and the user has only to supply a few parameters to make the process apply to the particular circumstances. Non-procedural specifications include the provision of keys for file sorting and the rules for copying, combining or blocking files.

Data Reference

I mentioned before that MADAM did not provide for the predefinition and symbolic naming of data elements. Instead data is referred to by its position in the data record image or storage area. A datum reference is thus, directly, an address which mentions the region containing the datum, the starting character of the datum relative to that region and the number of characters in the datum. This is a very natural way of specifying data to the data-handler who has before him the layout of the file or the format of a report.

Special Features

MADAM possesses several capabilities which are not found in the usual small-scale data management system. An indirect addressing capability enables general procedures to be established which can handle several different data elements or which can be used to control the formatting of variable length strings. There is the ability to specify general-purpose subroutines which can be called from different parts of the main specification. A library capability enables often-used specifications to be stored and called for execution by assigned name. The library function also permits partial specifications to be stored and assembled dynamically into a complete specification at execution time. Finally, there is the unusual capability to scan variable field data such as natural language text. Word separators can be defined by the user so that fields which are retrieved are appropriate whether the data is language, mathematical expression or computer code.

MADAM Applications

The following pages give brief accounts of some of the problems for which MADAM has been useful.

1. Data Documentation

A set of MADAM specifications known as DATADOX was written for the data management effort of the Bay Area Transportation Study Commission. Very large files of source data and derived data from questionnaires and other types of survey were collected. The files were processed by a large computer data management system called SPAN. The DATADOX specifications were written to process the SPAN specifications and, from the information contained in them, generate data dictionaries containing cross-indexed references to data elements, data lists, files, and so forth. By using

MADAM, BATSC analysts were able to solve one of the severe problems of any large-scale study, namely the accounting and documentation requirements for the wide range of statistical data used by the study.

2. Document Retrieval

The SURF system developed at SDC, is a series of MADAM specifications which provide the individual with a means of storing information about documents of interest to him, including such things as personal notes, letters and the like, traditionally the most difficult things to handle. The system operates as a service with subscribers. Subscribers are able to enter new data at intervals and are able to retrieve lists of relevant documents based on inquiries which are keyed on subject matter descriptors, author, title, date and so forth. While this is not so spectacular as an on-line retrieval system, the fact that many persons share in the use of a small inexpensive facility makes economically possible the personal, as opposed to the library, document retrieval system.

3. File Reformatting

It is frequently the case that data files exist in one format when the data they contain is required in an entirely different format. It is uneconomical and often impossible to collect the data again. MADAM is frequently used at SDC to reformat data. For example, a Department of Defense file containing information keypunched from a typical report form was radically altered for use with the LUCID system. MADAM was used to perform this task and without MADAM's text scanning ability the process might have been too expensive to be worth undertaking.

4. Tabulation

MADAM is used to prepare summary reports as an aid to management at SDC. It has also been used to produce scattergrams based on a statistical matrix. Because of limited core space this process, performed for a national professional organization, ran very slowly on the machine. In terms of total throughput time, however, it was probably faster than it would have been had it been necessary to code and check out programs for a large and powerful machine. The job, from presentation of the data to finished reports, required forty-eight hours.

The applications listed above are intended to show the wide range of problems for which MADAM may be used. It is only a small sampling of the actual uses that have been made, and, of course, a still smaller fraction of those for which the system is potentially useful.

An Example

This final section is for those hardy (and interested) enough to want to get the actual flavor of MADAM. The problem chosen is simpler than it would be in real life. It does, however, give an adequate view of how problems are attacked with MADAM and how simple and succinct the MADAM solution often turns out to be.

The problem concerns the updating of an inventory file. Three separate files are involved. They are:

1. Product file--contains the product number and the unit price of that product.

2. Inventory file--contains for each location a record for each product number on hand, together with the number of units and the total value.

3. Transaction file--this is a daily report file which contains a record for each product at each location for which a change in status has occurred. The change may either be a receipt of more items or a reduction in the number of items.

The requirement is to use the Transaction File to update the Inventory File, obtaining the unit price from the Product File. Describing the fields in the various files, the start is computed from zero. That is, the first character position of the record is position zero, the second, position one and so forth. This is how fields are referred to in MADAM and, to keep things consistent, these references will be used in describing the records of the various files.

The Product File

In real life this file would have many more fields, such as cost figures, blueprint numbers, patent numbers, Government specification numbere, name of product and so forth. For present purposes we predicate a file with only two fields--product code number and unit price. Furthermore, we assume that prices are in even dollars to avoid the complication of dealing with decimal points. The Inventory File, then, consists of twenty character records. The field 0 through 5 contains the product code number. The field 7 through 10 contains the unit price. The file is in product code number order.

PRODUCT FILE

Positions

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Record 1	0	0	0	0	1	7		0	0	1	0									
Record 2	0	0	0	0	2	1		0	0	0	3									
Record 3	0	0	0	0	4	0		0	1	0	5									
	-	-	-	-	-	-	-	-	-	-	-									
Record 200	0	1	3	4	0	5		0	2	1	6									

END OF FILE

The table above illustrates what such a file might look like printed out. The lowest numbered product code is 17 with a unit price of $10. The highest numbered code is 13405 with a unit price of $216.

<u>The Inventory File</u>

The Inventory File is sorted by location code and within location code by product number. Thus each record contains the quantity and total value of a particular product on hand at a particular location. The records are 30 character records. The location code is in positions 0 through 5, the product code number in positions 7 through 10, the quantity on hand in positions 12 through 17 and the total value in positions 19 through 28. The following table shows what this file might look like in printed report:

INVENTORY FILE

Location	Product	On Hand	Value
1	17	60	600
1	21	3006	9018
. . . .			
2	17	80	800
2	40	8	840
. . . .			
31	17	101	1010
31	40	7	735
. . . .			
31	13405	2	432

The Transaction File

The Transaction File consists of reports received daily from the various locations. Each reflects an inventory change at that location with respect to some product. The change may represent either an increase or a decrease in the number of units on hand. There may be more than one transaction at a location for a particular product. Of course, there will be products for which no transaction occurred on any particular reporting day. The Transaction File contains 20 character records and is unsorted. It has the following fields:

Positions	Contents
0 through 5	Location Code
7 through 10	Product Code Number
12	+ for accession or - for depletion
13 through 17	Units lost or gained

Strategy

The Transaction File is first matched against the Product File in order to calculate the value change resulting from the units lost or gained. The Transaction File is sorted on the product code number field which puts it in the same order as the Product File. Both of these files are inputs to the next process where each transaction is matched with a product. The number of units lost or gained from the Transaction File is multiplied by the unit price from the Inventory File. The product is stored in memory following the units lost or gained field and an output record is written containing the transaction information plus the computed value information. This new file is then sorted by product within location to put in the same relative order as the Inventory File. An updated Inventory File is now produced. Where no transactions have occurred the old record is copied. Where transactions have occurred with respect to existing inventory records, the inventory is modified by the units lost or gained and by the total value lost or gained. Where a transaction occurs where there was no corresponding record in the old file--representing a new product for the location--a new record is generated. No attempt is made here, as it would be in real life, to verify inputs to avoid, for example, ending up with a negative quantity on hand.

File Configuration

It is assumed that the machine being used has available six tape units. Units 1-4 are required for sorting and any tape other than the one being sorted would be destroyed if mounted on one of these units during the sort. To avoid this the Product File tape is mounted on Unit 5 and the Inventory File on Unit 6. The Transaction File, which is to be superseded in any case is mounted on Unit 2.

Notes on the Language

The MADAM Language is largely self-explanatory. Certain points, however, will help the reader follow the code.

a. Brackets consisting of two pairs of asterisks mark comments which are not translated. The experienced user seldom employs them, unless, as in this case, he wishes someone else to understand his specifications quickly.

b. A MADAM specification begins with a command (here only SORT and ABSTRACT occur), and ends with the term END.

c. Following the Command is the file specification, output first, then inputs. Each file is specified by a file number, unit number pair, e.g., ABSTRACT 1,2 1,3 1,5.

   This specifies that the output is file one of unit two and the two inputs are the first files of units three and five. The READ instruction specifies the particular input by its form, READ for the first input declared and READB for the second. The first input is read into region IN, the second into region INB.

d. When an IF statement is true, the instruction immediately following is executed. When it is false the next labelled instruction is executed.

The Job

Here, then, is a set of MADAM specifications, four in all, which constitute a job, the job being to update the Inventory File from the daily transactions.

```
**INVENTORY UPDATE**

**MATCH TRANSACTION TO PRODUCT**

SORT **TRANS. FILE** 1,2

BY **KEY, PRODUCT NUMBER** 7,4

**MAXIMUM RECORD SIZE** 20

END


**OBTAIN VALUE CHANGE AND GENERATE

NEW TRANSACTION FILE**
```

```
        ABSTRACT **NEW TRANS FILE** 1,3
    **INPUT 1, PRODUCT FILE** 1,5
    **INPUT 2, TRANS FILE** 1,2

    READ **PRODUCT RECORD**
1.  READB **TRANSACTION RECORD**
2.  IF IN 0,4 LS INB 7,4 **NO MATCH**
    READ **NEXT PRODUCT RECORD**
    DO 2 **TRY MATCH AGAIN**
3.  COMPUTE INB 13,5 * IN 5,6
    ** UNITS CHANGE TIMES UNIT VALUE **
      = INB 19,10  ** TOTAL VALUE FIELD **
    WRITE INB 0,30  ** AUGMENTED RECORD **
      DO 1  ** FETCH NEXT TRANSACTION **
              END
```

(Note that process terminates automatically when end of file is read on either input.)

```
** MATCH NEW TRANSACTION FILE AGAINST
  INVENTORY FILE **

SORT ** NEW TRANS FILE ** 1,3
  BY ** MAJOR KEY, LOCATION ** 0,6
```

```
                AND ** MINOR KEY, PRODUCT ** 7,4
                ** MAX RECORD SIZE ** 30
                     END

                ** PRODUCE UPDATED INVENTORY **

                ABSTRACT ** NEW INVENTORY ** 1,2
                  ** INPUT 1, OLD INV. ** 1,6
                  ** INPUT 2, NEW TRANS ** 1,3

                READ ** INVENT. RECORD **

1.              READB  ** TRANS RECORD **
                IF ENDB ** NO MORE TRANS ** DO REST

2.              IF IN 0,6 LS INB 0,6 ** NOT THIS LOCATION **
                OR IN 7,4 LS INB 7,4 ** NOT THIS PRODUCT **
                WRITE IN 0,30 ** COPY OLD RECORD **
                READ ** NEXT INV RECORD ** DO 2

3.              IF IN 0,6 GR INB 0,6 ** NEXT LOCATION **
                OR IN 7,4 GR INB 7,4 ** NEXT PRODUCT **
                WRITE INB 0,30 ** NEW ENTRY ** DO 1
```

```
4.          ** RECORDS MUST MATCH **
            IF INB 12,1 EQ (-) ** UNITS LOST **
            COMPUTE IN 12,6 - INB 13,5 = IN 12,6 ** UNITS **
            COMPUTE IN 19,10 - INB 19,10 = IN 19,10 ** VALUE **
            WRITE IN 0,30
            DO 1 ** GET NEXT TRANS **

5.          ** THIS IS A GAIN **
            COMPUTE IN 12,6 + INB 13,5 = IN 12,6
            COMPUTE IN 19,10 + INB 19,10 = IN 19,10
            WRITE IN 0,30
            DO 1
REST.       ** COPY PART OF OLD INV NOT REPRESENTED
                    IN TRANS FILE **
            WRITE IN 0,30 ** CURRENT RECORD **
            READ ** NEXT INV RECORD **
            DO REST ** CONTINUE TO END OF FILE **
                    END
```

Slim and Trim

Lest the casual observer should think the above specification set is long and laborious, I have reproduced the same instructions below without the commentary.

```
SORT 1,2 BY 7,4,20 END
ABSTRACT 1,3 1,5 1,2
   READ
```

```
    1. READB
    2. IF IN 0,4 LS INB 7,4  READ DO 2
    3. COMPUTE INB 13,5 * IN 5,6 = INB 19,10
       WRITE INB 0,30 DO 1  END

   SORT 1,3 BY 0,6 AND 7,4,30 END
   ABSTRACT  1,2  1,6  1,3
      READ
   1. READB IF ENDB DO REST
   2. IF IN 0,6 LS INB 0,6 OR IN 7,4 LS INB 7,4
      WRITE IN 0,30 READ DO 2
   3. IF IN 0,6 GR INB 0,6 OR IN 7,4 GR INB 7,4
      WRITE INB 0,30 DO 1
   4. IF INB 12,1 EQ (-)
      COMPUTE IN 12,6 - INB 13,5 = IN 12,6
      COMPUTE IN 19,10 - INB 19,10 = IN 19,10
      WRITE IN 0,30 DO 1
   5. COMPUTE IN 12,6 + INB 13,5 = IN 12,6
      COMPUTE IN 19,10 + INB 19,10 = IN 19,10
      WRITE IN 0,30 DO 1
REST. WRITE IN 0,30 READ DO REST    END
```

BIBLIOGRAPHY

1. Crossley, W. O., "The MADAM System," System Development Corporation, TM-2198/002/00, December 1965.

2. Crossley, W. O., "Relating Street Addresses to Census Tracts: An Application of MADAM," System Development Corporation, SP-(L)-1902, December 1964.

3. Franks, E. W., "The MADAM Language for the IBM 360/30 Computer," System Development Corporation, TM-3555/000/00, June 1967.

4. Kibbee, J. M., and V. V. Almendinger, "Bug Area Transportation Study Commission Information System: Data Description and Documentation," System Development Corporation, TM-2690/000/00, December 1965.

5. Wallace, E. M., "A User's Guide to SURF: Support of User Records and Files," System Development Corporation, TM-2913/000/00, June 1966.

DOCUMENT CONTROL DATA - R&D

1. ORIGINATING ACTIVITY (Corporate author) System Development Corporation Santa Monica, California	2a. Unclassified 2b.

3. REPORT TITLE

The MADAM System: Data Management with a Small Computer.

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)

5. AUTHOR(S) (Last name, first name, initial)

Franks, Emory W.

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
8 September 1967	16	5

8a. CONTRACT OR GRANT NO. U.S. Government Contracts b. PROJECT NO. c. d.	9a. ORIGINATOR'S REPORT NUMBER(S) SP-2944 9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)

10. AVAILABILITY/LIMITATION NOTICES

Distribution of this document is unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSOR

13. ABSTRACT

This paper discusses the uses of the MADAM system for data management tasks on the small, free-standing computer such as the IBM 1401 or IBM 360/30. It is largely nontechnical in approach and is addressed to managers rather than to computer personnel. An example of a problem and its solution is given to demonstrate the speed and simplicity of MADAM usage.

A-2735

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
MADAM System Data management Small computers						